Discrete Math in CS	**Problem Set 6**
CS 280 Fall 2005	**Due October 28, 2005**

Reading: Rosen, Section 5.3.

---

(1) Let's continue thinking about the model of randomized load balancing that we discussed in Question (1) of the previous problem set. As before, we have n identical computing servers, and n jobs that need to be performed. To deal with the fact that the global state of the system is too hard to maintain, we assign each job independently and uniformly at random to one of the servers. That is, the jobs arrive in sequence, and each is assigned uniformly at random to one of the servers, independent of the assignment of previous jobs.

(a) Let $Z(n)$ be the expected number of machines that do not receive any jobs, so that $Z(n)/n$ is the expected fraction of machines with nothing to do. What is the value of the limit $\lim_{n\to\infty} Z(n)/n$? Give a proof of your answer.

(b) Suppose that machines are not able to queue up excess jobs, so if the random assignment of jobs to machines sends more than one job to a machine M, then M will do the first of the jobs it receives and reject the rest. Let $R(n)$ be the expected number of rejected jobs; so $R(n)/n$ is the expected fraction of rejected jobs. What is $\lim_{n\to\infty} R(n)/n$? Give a proof of your answer.

---

(2) In class, we saw a simple distributed protocol to solve a particular contention-resolution problem. Here is another setting in which randomization can help with contention-resolution, through the distributed construction of a *conflict-free set*.

Suppose we have a system with n processes. Certain pairs of processes are in *conflict*, meaning that they both require access to a shared resource. In a given time interval, the goal is to schedule a large subset S of the processes to run — the rest will remain idle — so that no two conflicting processes are both in the scheduled set S. We'll call such a set S *conflict-free*.

We'd like a simple method for selecting a large, conflict-free set of processes without centralized control: each process should communicate with only a small number of other processes, and then decide whether or not it should belong to the conflict-free set S.

We will suppose for purposes of this question that each each process is in conflict with exactly d other processes.

(a) Consider the following simple protocol.

> Each process P_i independently picks a random value x_i; it sets x_i to 1 with probability $\frac{1}{2}$ and set x_i to 0 with probability $\frac{1}{2}$. It then decides to enter the set

> S if and only if it chooses the value 1, and each of the processes with which it is in conflict chooses the value 0.

Prove that the set S resulting from the execution of this protocol is conflict-free, and give a formula for the expected size of S in terms of n (the number of processes) and d (the number of conflicts per process).

(b) The choice of the probability $\frac{1}{2}$ in the protocol above was fairly arbitrary, and it's not clear that it should give the best system performance. A more general specification of the protocol would replace the probability $\frac{1}{2}$ by a parameter p between 0 and 1, as follows:

> Each process P_i independently picks a random value x_i; it sets x_i to 1 with probability p and set x_i to 0 with probability $1 - p$. It then decides to enter the set S if and only if it chooses the value 1, and each of the processes with which it is in conflict chooses the value 0.

In terms of the parameters n and d, give a value of p so that the expected size of the resulting set S is as large as possible. Give a formula for the expected size of S when p is set to this optimal value.

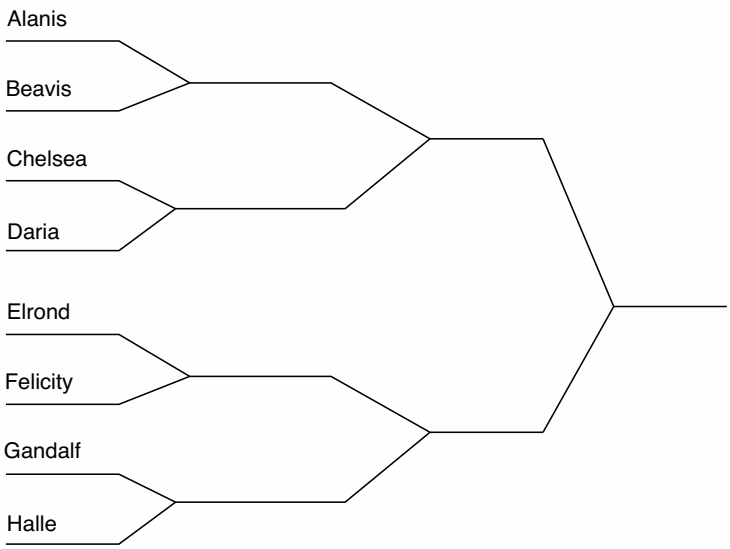

Figure 1: An elimination tournament with just the initial contestants, before any matches have been played.

(3) Your friends never expected their video game *Gran Turismo 7: Upson Hall* to become the wild success that it has, but they're already in the process of organizing the first-ever *GT7:UH* tournament over the Internet.

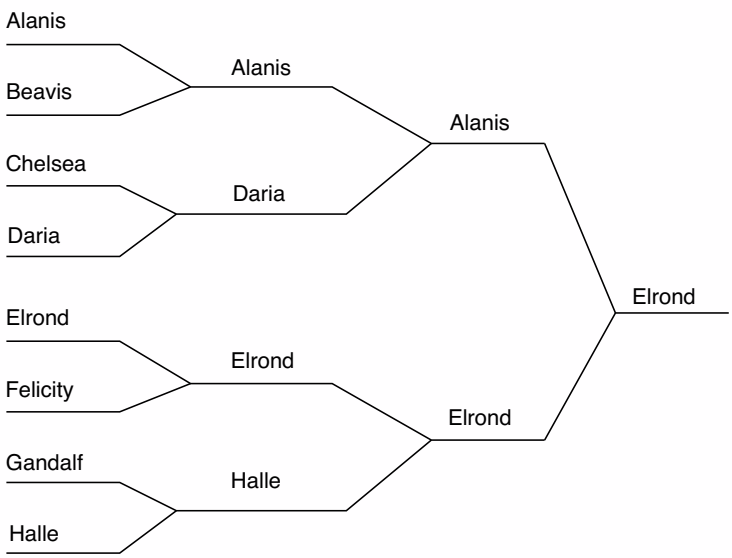

Figure 2: The results of the elimination tournament, after all matches have been played.

This is a k-round, single-elimination tournament. In other words, 2^k contestants start out at the beginning, and in each round the current contestants play each other in specified pairs, with the winners moving on to the next round. An example with $k = 3$ and $2^3 = 8$ initial contestants is depicted in Figures 1 and 2. One can also picture this as a complete binary tree with the initial contestants at the leaves, and each internal node corresponding to a match-up of two surviving contestants. (The NCAA basketball tournament is a well-known example of this format, with $k = 6$.)

Given the intense interest from the GT7 fan base, your friends want to keep things engaging for all the people who weren't selected to be contestants in the actual tournament. So on the side, they organize a betting pool, asking people to predict the outcomes of all the matches in the tournament.

Specifically, this means that you get a copy of a blank table with just the initial contestants filled in (as in Figure 1), and you're asked to fill in all the entries for the subsequent rounds. Crucially, all entries must be filled in before any matches are played. The filling-in should be consistent, in that if you write a person's name as a winner in round j, you should also have guessed that they're a winner in the previous round $j-1$. Your *score* is the number of entries for which the filled-in guess turns out to be the same as the name of the person who actually ended up in that entry. The goal is to get as high a score as possible. (For example, suppose you predict Alanis, Chelsea, Elrond, and Gandalf as the winners in the first round; then Alanis and Gandalf as winners in the second round; and then Alanis in the third (final) round. The number of correct predictions here would be three (two in the first round, and one in the second round).)

Let's consider how well we'd expect someone to do at guessing these results, given that they don't know what's going to happen. To be concrete, we'll model the uncertainty

about the outcome of each match by assuming (simplistically) that each match's outcome is determined by an independent fair coin flip. (So each contestant is equally likely to win and advance to the next round, where their fate will be determined by the next coin flip).

(a) Let X denote the random variable equal to the number of correct predictions, when a table is first filled in with guesses, and then the outcome of each match is determined by independent, fair coin flips. What is the expected value of X? Give an explanation for your answer.

(b) Let $p(j)$ denote the probability that *any* of a person's guesses for the entries in round j turn out to be correct. That is, we look at all their guesses for the contestants in round j, and see if any of these contestants actually made it to round j. (So, in our previous example with three correct guesses, the person had at least one correct guess in the first round and at least one correct guess in the second round, but none in the third round.) What is

$$\lim_{k\to\infty} p(k/2)?$$

What is

$$\lim_{k\to\infty} p(2k/3)?$$

(In other words, what is the chance of still having a correct guess at the halfway and two-thirds points in the tournament, respectively, as k gets large?) Give explanations for your answers.